REMARKS

In an office action dated April 20, 2004, the Examiner rejected claims 1-4, 7-9, and 12-15 under 35 U.S.C. §103(a) as obvious over Kleinsorge et al. (US Patent 6,247,109) in view of Larson (US Patent 6,115,705). Claims 5, 6, 10, 11 and 16 were objected to as dependent on rejected base claims, but otherwise indicated to contained allowable subject matter.

Claims 5, 6, 10, 11 have been re-written in independent form, and are therefore allowable. Claim 12 has been amended to incorporate the limitations previously contained in dependent claim 16 (now cancelled), and as amended claim 12 is therefore also allowable. Claims 13-15 are dependent on claim 12, and allowable for the same reasons.

Claim 1 has been amended to clarify that the step of comparing processor resource assignments is performed automatically, i.e., is performed by the system, as opposed to the user manually invoking a re-optimization of the query. This change is unnecessary for independent claim 7, since claim 7 inherently recites functions performed by a machine. Applicants respectfully traverse the rejections of the claims.

Applicants' invention relates to the optimization of database queries in a logically partitioned environment. As is well known, any given logical database query can be executed in a variety of ways. Conventional query optimizers are available to choose an optimal execution strategy for a given query. In some cases, this optimal strategy may depend on the system itself, i.e. the amount and type of each system resource available, and in particular, the processor resource available.

If a single physical system is logically partitioned, then the system resources are assigned to the various partitions to create multiple virtual computer systems. Each user task executes in a respective partition, and from the perspective of the task, it appears that its virtual computer system is limited to the resources assigned to that logical partition. In some forms of logical partitioning, it is possible to dynamically re-assign resources to different logical partitions. This might be performed, e.g., because certain scheduled tasks running at particular times require additional resource.

A database query, like any other user task, executes in one of the logical partitions. A query optimizer will therefore optimize the query based on the resources assigned to some logical partition, generally the logical partition in which the optimizer executes. Such an optimized query can be saved for later re-use, or for use on a different logical partition. If such a saved query is subsequently executed in a logical partition having a different resource assignment from the one for which it was optimized (either because the resource assignment changed, or because the query is executed in a different logical partition), then the execution strategy may be sub-optimal.

In accordance with applicants' invention, the processor resource assignment for which the query was originally optimized is compared with the processor resource assignment of the current logical partition of execution, just before the query is executed. If there has been a significant change to processor resource assignment, the query is automatically re-optimized for the new resource assignment.

Kleinsorge discloses a logically partitioned computer system having various capabilities to dynamically manage the logical partitioning environment, including in particular the capability to dynamically re-assign processor resources. However, *Kleinsorge* discloses nothing about database managment, and in particular does not disclose database query optimization dependent on a logical partitioning environment.

Larson discloses a technique for database management and query execution. In accordance with *Larson*'s technique, a database query processor divides the real memory space available to the query task into multiple "partitions", each partition holding some subset of database records according to a hash function of some value to be sorted. It performs aggregation and sorting of the records within the "partitions" separately. By thus allocating separate groups of records to separate "partitions" of memory, the query task can separately sort and process each group individually, reducing the amount of processing time required for the task as a whole.

It should be understood that *Larson*'s "partitions" of memory are not logical partitions in the sense that term is used in applicants' specification and claims. A "logical partition" is an isolation of system resources for performance of tasks. Any individual task recognizes only a single logical partition, and can only utilize the resources allocated to that logical partition. *Larson*'s memory "partitions" are simply divisions of the memory heap space available to the query task. In this respect, *Larson*'s query task is no different from any other task executing in a computer system. Any task divides the available real memory up into portions useful to the executing task. Whether these portions are called "data structures", "arrays", "blocks", or by some other name, the task is simply dividing up and allocating its available memory. This is not the same as logical partitioning, in which system resources are divided into "partitions", and any given task sees only what is allocated to its respective partition.[1]

Applicants' claim 1 recites in part:

> 1. A method for database query optimization ... comprising the steps of:
> defining a plurality of logical partitions ... each ... having a respective processor resource assignment, *wherein each task ... is assigned to a respective one of said logical partitions* and wherein the definition of a plurality of logical partitions may be dynamically altered;

[1] To be precise, high-level tasks or user tasks, specifically including database query tasks, are limited to the resources allocated to a particular partition. At some low level, a "hypervisor" or equivalent task manages logical partitioning itself, and this task does indeed see the resources of the entire system.

> defining a database query;
> constructing a first search strategy for said database query, said first search strategy being dependent on a first processor resource assignment ... ;
> invoking said database query for execution in a first logical partition ... ;
> *automatically comparing a second processor resource assignment to said first processor resource assignment*, said second processor resource assignment being associated with said first logical partition at the time said invoking said database query for execution step is performed; and
> automatically constructing a second search strategy dependent on said second processor resource assignment, *said step of automatically constructing a second search strategy being performed dependent on the results of said comparing step.* [emphasis added]

Claim 7 contains analogous limitations.

Applicants do not challenge the combination of *Kleinsorge* and *Larson.* But just what does such a combination teach? The combination amounts only to executing a query task such as disclosed in *Larson* in a logical partition such as disclosed in *Kleinsorge.* This combination fails to meet the limitations of applicants' claim 1. In particular, it fails to teach or suggest the steps of "comparing a second processor resource assignment to a first processor resource assignment...", and "automatically constructing a second search strategy..." (re-optimizing the database query) based on the results of the comparison.

The Examiner cites a passage from *Kleinsorge* which discloses that a processor can be migrated from one logical partition to another, preserving its state. In a general sense, this action might be viewed as "comparing a second processor resource assignment to a first processor resource assignment", but this ignores the remainder of the limitation. The first processor resource assignment is associated with a database query. And the following step uses the results of this comparison to determine whether to re-optimize the database query. *Kleinsorge*'s comparison, if it is performed at all, is only used to migrate and initialize a processor state, not to re-optimize a database query. An while *Larson* discloses the use of database queries, there is nothing that would teach or suggest the selective re-optimization of a database query depending on the results of a comparison between processor resource assignments at the time of the original optimization and at the time of execution.

The problem addressed by applicants' invention is a problem which arises as a result of the use of database query optimizers in loigcally partitioned systems which can be dynamically re-partitioned. I.e., the problem is that the assumptions upon which a query was optimized might no longer be true due to dynamic re-partitioning. This particular problem is not mentioned anywhere in either reference. While the use of database queries and logical partitioning separately is shown, and the suggestion to perform database queries in logically partitioned systems may be inferred, there is nothing in the references that suggest the particular problem addressed by this invention, or the solution proposed by applicants. For all of the reasons stated, claims 1 and 7 are neither taught or suggested by the combination of *Kleinsorge* and *Larson.*

In view of the foregoing, applicants submit that the claims are now in condition for allowance and respectfully request reconsideration and allowance of all claims. In addition, the Examiner is encouraged to contact applicants' attorney by telephone if there are outstanding issues left to be resolved to place this case in condition for allowance.

Respectfully submitted,

PAUL R. DAY, et al.

By: [illegible]
Roy W. Truelson
Registration No. 34,265

Telephone: (507) 289-6256

Docket No.: ROC920010160US1
Serial No.: 10/001,667